Middle East Technical University   Department of Computer Engineering

CENG 232

Logic Design

Spring '2019-2020

Lab 3

Part 1 Due Date: 5 April 2020, Sunday, 23:59
Part 2 Due Date: 19 April 2020, Sunday, 23:59
No late submissions

1 Introduction

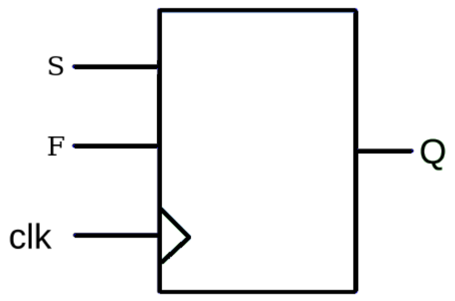
This assignment aims to make you familiar with Verilog language, related software tools, and the FPGA boards. There are two parts in this assignment. The first part is a Verilog simulation of an imaginary flip-flop design, which you are required to implement and test on your own. The second part consists of implementation and simulation (by testbenches) of A Multi-Agent Systems (MAS) Application system.

2 Part 1: SF - Flip Flop (Warming - 25 pts)

You are given a specification of a new type of flip-flop, and a new chip that uses the flip-flop. This is an individual part. Your task is to implement these in Verilog, and prove that they work according to their specifications by simulation (this part will only be tested with testbenches not with FPGAs).

2.1

Implement the following SF flip-flop in Verilog with respect to the provided truth table. An SF flip-flop has 4 operations when inputs S and F are: 00 (set to 0), 01 (set to 1), 10 (no change), 11 (complement). Please note that the SF Flip-Flop changes its state only at rising clock edges.



2.2

Implement the following chip that contains one SF flip-flop which has output Q.

Use the following module definitions for the modules:

```
module sf(input S, input F, input clk, output reg Q)
module ic1500(input B, input A, input X, input clk, output D0, output D1, output Q, output Z)
```

Table 1: SF - flip-flop truth table.

S	F	Q	Qnext
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

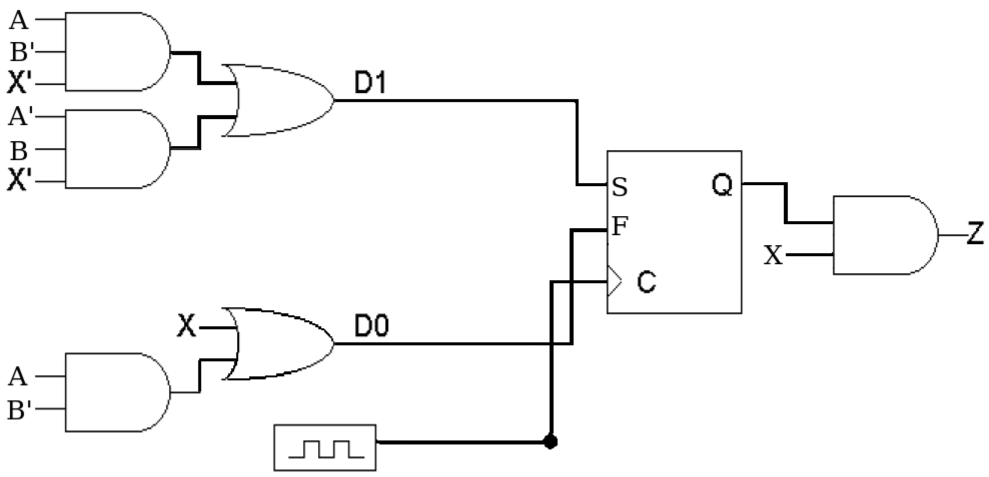


Figure 1: The figure shows inside of the ic1500 module (connections).

In the Figure-1, inputs of the ic1500 module are B, A and X; and the output of the ic1500 module is Z. All B's, all A's and all X's are coming from single sources (there is one A, one B and one X) but to make it look simpler we did not show the cable connections. A', B' and X' are the negated versions of the A, B and X. D0 and D1 are the inputs of SF flip-flop. C is the clock, and Q is the output of SF flip-flop and the Z is the output of the ic1500 module.

2.3 Simulation

A sample testbench for sf-flipflop module will be provided to you. It is your responsibility to extend the testbench, and also to write a testbench for ic1500 module.

2.4 Deliverables

- Implement both modules in a single Verilog file: lab3_1.v. Do NOT submit your testbenches. You can share your testbenches on the newsgroup.
- Submit the file through the Odtuclass system before the given deadline. **5 April, 2020, 23:59**
- This part is supposed to be done individually. Any kind of cheating is not allowed (code sharing with your friends, taking codes from internet etc.).
- **Odtuclass Discussions:** You should follow the odtuclass for discussions and possible updates on a daily basis.

3 Part 2: A Multi-Agent Systems (MAS) Application (75 pts)

Your task is to implement these in Verilog, and prove that they work according to their specifications by simulation (this part will only be tested with testbenches not with FPGAs). This is an individual part, so should not share any code with your friends, and you should not take any code from internet, all the work provided should be your own work.

3.1 Problem Definition

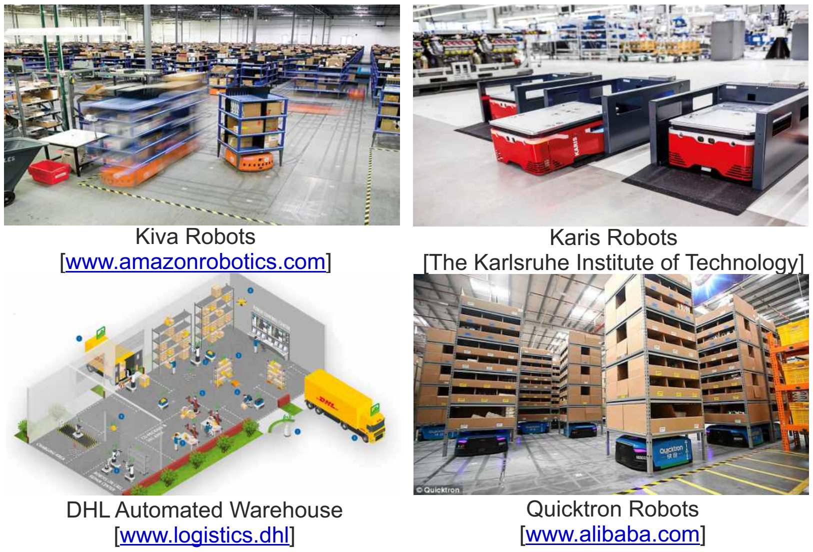


Figure 2: Example application areas of multi-agent systems. Automated warehouses are started to be used by many companies.

Technology is improving and in the near future, intelligent agents will take place of the humans in many professions. One of the research areas that affect this technology most is multi-agent systems. In this homework, you will make an introduction to the multi-agent systems (MAS) area. MAS aims to use multiple intelligent agents for the problems that are difficult or impossible to solve with an individual agent. The intelligence of the agents may be satisfied with search algorithms, reinforcement learning, etc.

MAS has various application areas from warehouses (Figure-2), games, autonomous swarms and military applications (more can be added).

In this homework, you will designate simulation for a single robot to follow its movements. This will make a baseline study to develop it further and generate a hardware simulation for a multi-agent platform. There will be a single agent in your application. It will operate in a 3x3 grid world. By each clock cycle, it will operate a new action (to move a new cell or to stand still). The possible actions that your robot can do are shown in Figure-3-a, it can move to neighboring locations and can only make one move at a single clock cycle. The possible actions that your agent can do are enumerated (including wait action) and shown in Figure-3-b. Also your grids all cells and their coordinates are shown in Figure-3-c.

Your work in this homework is to execute new commands (that tell your agent to move an adjacent cell or standstill) with your agent at each clock cycle, calculate and show the current coordinates of your agent. Calculate and show the total cost of your agent to move from its beginning location to its current location and to calculate and show the total time elapsed in the simulation.

As you are a systematically working, computer engineer wanna be; you generated your own specifications before implementing the system:

1. Each **command** is a 4-bit number.
2. Each command defines a movement for the agent.
3. Each movement takes a unit time (one clock cycle).
4. Agent can only **move to its neighbor cells** (agent can move from (0,1) to (0,2) as (0,2) is a neighbor cell, but agent can not move from (0,1) to (2,2) as they are not neighbor cells) or **choose to stand still**.

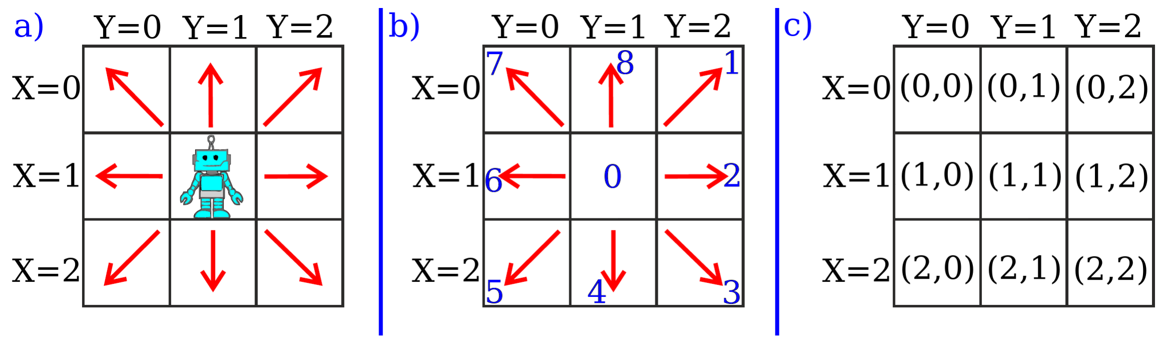


Figure 3: 3x3 grid that will be used in this homework; a) presents the possible actions that your agent can execute, b) presents the possible actions in enumerated form, c) presents the coordinates of the grid cells.

5. Agent can move to right (2), left (6), up (8), down (4), up-right (1), up-left (7), down-right (3), down-left (5) cells or can stand still (0) for that time step (see Figure-3-b).
6. If command is equal to:
   - **0 (0000)** → the agent will execute a wait action. (The agent will stand still.)
   - **1 (0001)** → the agent will execute a diagonal move to up-right neighbor cell. If there are no more up-right cell agents will standstill.
   - **2 (0010)** → the agent will execute a move to right neighbor cell. If there are no more right cells, the agent will move to the leftmost cell of the same row (make a circular move).
   - **3 (0011)** → the agent will execute a diagonal move to down-right neighbor cell. If there are no more down-right cells agents will standstill.
   - **4 (0100)** → the agent will execute a move to down neighbor cell. If there are no more down cells, the agent will move to the upmost cell of the same column (make a circular move).
   - **5 (0101)** → the agent will execute a diagonal move to down-left neighbor cell. If there are no more down-left cells agent will standstill.
   - **6 (0110)** → the agent will execute a move to left neighbor cell. If there are no more left cells, the agent will move to the rightmost cell of the same row (make a circular move).
   - **7 (0111)** → the agent will execute a diagonal move to up-left neighbor cell. If there are no more up-left cells agents will standstill.
   - **8 (1000)** → the agent will execute a move to up neighbor cell. If there are no more up cells, the agent will move to the down most cell of the same column (make a circular move).
   - If the command is larger than 8 then the command will be **discarded**. The agent will do nothing and the cost will be 0 (no cost).
7. The cost of command 0 is 1 unit (standing still costs 1).
8. The cost of commands 2,4,6 and 8 (right, down, left and up) is 2 units (If the agent will do a circular move it will cost 2 units (not additional, just the total cost of that action will be 2)).
9. The cost of commands 1, 3,5 and 7 (diagonal moves) is 3 units. If the agent is at a border location and can not execute the diagonal move then that agent will stand still and the cost of it is 1 unit (not 3 units).
10. The commands 9 to 16 will be discarded and they won't cost anything.
11. After executing each command, total-time should be increased by one.
12. You have two **modes** in this system. The **reset-mode** is active when the **mode** option is 1, **working-mode** is active when the **mode** option is 0.
13. If the system is in the **reset-mode**, you should set the total-cost, total-time and the coordinates of the agent to 0. And you should activate the warning led (make it 1).

14. If the system is in the **working-mode**, with each rising edge of the clock, the agent should execute the current command. The total-cost and the coordinates of the agent should be updated. (The warning led must be turned off).
15. Initially, the agent is placed at (1,1) cell, the total-cost and total-time variables will be set to 0.
16. Don't forget, after the reset action the total-cost, total-time and the coordinates of the agent to 0. This means that the agent will be placed at (0,0) location after the agent goes to **working-mode** from **reset-mode**.
17. The total-cost and the total-time will be shown in decimal form with two digits (One digit is total_time1 (most significant) and the other digit is total_time0 (least significant), same applies to total_cost).
18. The coordinates of the agent will be two-bit numbers between 0 (00) and 2 (10).
19. The total-cost and the total-time varies between 0 and 20. (2 is one digit and 0 is another digit)
20. If the total-cost (or total-time) exceeds 20, it should be returned to 0. (20 is not included, after 19 it returns to 00. For the total cost, after the cost is 19 and the new cost is 2, then the next cost will be 01. Similarly 17 + 3 will be 00, 19 + 3 will be 02 etc.)

3.2

You should test your implementations with test benches. You should generate your own testbenches, no testbench will be provided to you.

3.3 Sample Input/Output

- The values in **Current State** column, which are seperated by "**,**" are defined as: **command**, **mode**, **coord_x (2 bits)**, **coord_y (2 bits)**, **total_cost1**, **total_cost0** , **total_time1**, **total_time0** respectively. (**total_time1** is the most significant digit of number of **total_time**, **total_time0** is the least significant digit of **total_time**. Same is applied for **total_cost**)
- The values in **Next State** column, which are seperated by "**,**" are defined as: **coord_x (2 bits)**, **coord_y (2 bits)**, **total_cost1**, **total_cost0**, **total_time1**, **total_time0**, **warning**, respectively.

Table 2: Sample inputs and outputs.

current state	CLK	next state
0000, 0, 01, 01, 0, 0, 0, 0	↑	01, 01, 0, 1, 0, 1, 0
0001, 0, 01, 01, 0, 1, 0, 1	↑	00, 10, 0, 4, 0, 2, 0
0111, 0, 00, 10, 0, 4, 0, 2	↑	00, 10, 0, 5, 0, 3, 0
0010, 0, 00, 10, 0, 5, 0, 3	↑	00, 00, 0, 7, 0, 4, 0
0011, 1, 00, 00, 0, 7, 0, 4	↑	00, 00, 0, 0, 0, 0, 1
0011, 0, 00, 00, 0, 0, 0, 0	↑	01, 01, 0, 3, 0, 1, 0
0111, 0, 01, 01, 1, 9, 1, 3	↑	00, 00, 0, 2, 1, 4, 0

3.4 Input/Output Specifications

- **command** represents 4-bit code.
- **CLK** is the clock input for the module.
- **mode** is used to indicate whether the system is in the reset-mode or in the working-mode.
  mode = 0 ⇒ with each rising edge of the clock, the agent should execute the current command. The total-cost and the coordinates of the agent should be updated (working-mode).
  mode = 1 ⇒ the total-cost, total-time and the coordinates of the agent should be set to 0. And you should activate the warning led (reset-mode).
- **total_time1**, **total_time0** and **total_cost1**, **total_cost0** shows the total-cost and total-time values of the simulation. Where 1 is the most significant digit and 0 is the least significant digit.

- **coord_x** and **coord_y** are two bit numbers that determine the current place of the agent. For example, coord_x = 00 and coord_y = 01 means that agent is at (0,1) location.

- **warning** shows the warning situations.
  warning = 0 ⇒ no warning occured (in working-mode).
  warning = 1 ⇒ warning occured (in reset-mode).

Table 3: Inputs and output variables

Name	Type	Size
word	Input	4 bits
Clock (CLK)	Input	1 bit
mode	Input	1 bit
total_time1	Output	8 bits
total_time0	Output	8 bits
total_cost1	Output	8 bits
total_cost0	Output	8 bits
coord_x	Output	2 bits
coord_y	Output	2 bits
warning	Output	1 bit

3.5 Deliverables

- Implement your module in a single Verilog file: lab3_2.v. Do NOT submit your testbenches.You may share your testbenches on the newsgroup.
- Submit the file through the Odtuclass system before the given deadline. **April 19, 2020, 23:59**
- **This is not a group part!!! All the work will be done individually. Any kind of cheating is not allowed.**
- **Odtuclass Discussions:** You should follow the odtuclass for discussions and possible updates on a daily basis.